ANNEX 1.

```
Needs["RingFunctionsDiffEncoded`"]

Names["RingFunctionsDiffEncoded`*"]

{AllGoldSequences, AutocorrelationSequence, AutomorphismSigma, CodingTransform,
 CodingTransformNew, CrosscorrelationSequence, CyclicMultiplicativeGroup,
 DropLeadingZeros, GoldSequence, GraeffeMethod, InitialConditions, MinimumPoly,
 ModuloMultplication, PolynomialMultiplication, PossibleDivisors, RingDivision,
 RingPower, SequenceGenerator, SpecifiedGoldSequences, TraceRepresentation,
 TupleRepresentation, TwoAdicExpansion, UnitsRing, ZeroPad, ZeroSequences, τ, σ}

Needs["LaurentFunctions`"]

RuleDelayed::rhs : Pattern t_ appears on the right-hand side of rule
   PhaseAngle[L_][t_] :> (PhaseAngle[L][t_] = Module[{x1, x2, x3, x4, x5, x6}, <<1>>]).

Needs["LaurentNotationTest`"]

Needs::nocont : Context LaurentNotationTest` was not created when Needs was evaluated.
```

Information on the functions used can be obtained using help.

```
Names["LaurentFunctions`*"]

{AKN, AlphaKI, ANKInitialStateSetUp, BT, FiltPulse, h, hFiltered, InitialState, J,
 LaurentC, LaurentLK, LaurentS, M, ModulatingPulse, ModulationIndex, Modulator,
 NumberOfCurves, PhaseAngle, PhaseAngleFast, Receiver, ReceiverProper, S,
 SamplingInterval, StartingQuadrant, SyncSample, T, C, Φ, ψ}
```

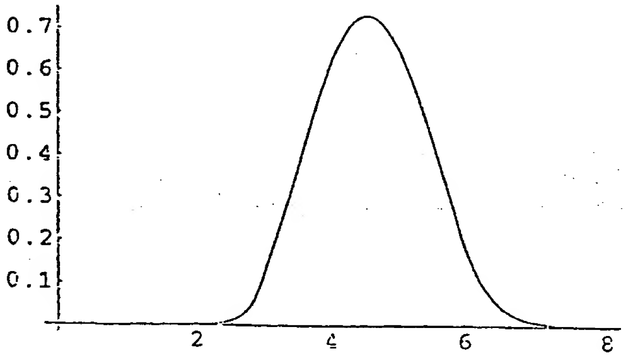
$T := \frac{3}{812500}$ GSM value of T

```
BT := 0.3
```

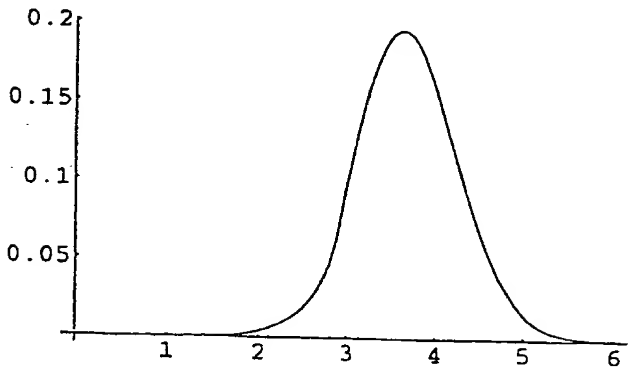
$\text{ModulationIndex} := \frac{1}{2}$

```
<< ModulatorData.m;

<< OptimalPulseShapes.m;

Plot[OptPulse[L][0][t], {t, 0 , 8}]
```

Example of optimal pulse shape.



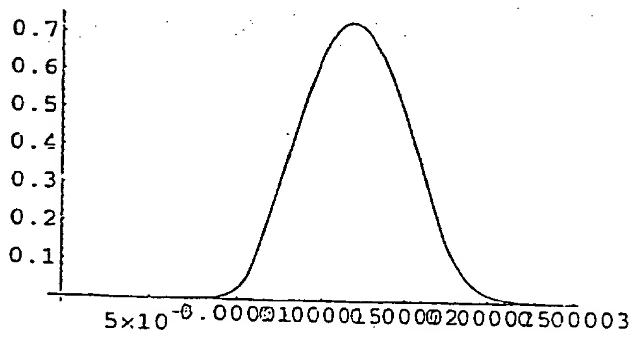
```
- Graphics -

Table[
Plot[OptPulse[L][1][t], {t, 0 , 6)]
```



```
- Graphics -

T

   3
------
812500
```

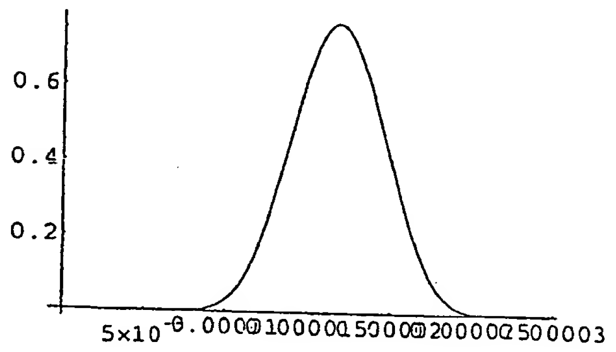
The unit of time is $T=1$ for OptPulse.We scale the Pulses to $T=\frac{3}{812500}$ for the unit of time.

```
OptPulseScaled[8][0][t_]  := OptPulse[L][0][t / T]
OptPulseScaled[8][1][t_]  := OptPulse[L][1][t / T]

Plot[OptPulseScaled[L][0][t], {t, 0 , 8 T}]
```



```
- Graphics -


RandomBitSeq

{1, 1, -1, -1, -1, 1, 1, -1, 1, -1, 1, -1, 1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, 1, 1, 1,
 -1, 1, -1, -1, 1, -1, -1, 1, 1, 1, 1, -1, 1, -1, 1, -1, 1, -1, -1, 1, 1, 1, 1, -1,
 1, -1, 1, -1, 1, -1, 1, 1, -1, -1, 1, -1, 1, -1, 1, 1, 1, 1, -1, -1, -1, 1, -1,
 1, 1, 1, -1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, 1, -1, 1, -1}
```

```
Plot[FiltPulse[L][0][t], {t, 0, 8 T}]
```



```
- Graphics -
```

The Gold Sequence Set

We generate the sequences using the method specified by Serdar Boztas and P Vijay Kumar in Ref [1]. The numbering of the sequences is the one used in the paper. We generate a small subset of the sequences. There are $2^{10}+1$ sequences with the quaternary polynomial used. Given any binary primitive polynomial, we can generate the corresponding quaternary polynomial.

```
Goldseqlist =
 SpecifiedGoldSequences[{1, 3, 2, 1, 0, 3, 0, 0, 1}][{1, 2, 3, Last - 1, Last}];
```

The last sequence of seqlist has nice autocorelation properties.

The last sequence is in fact a m–sequence of length 1023 bits.

```
Goldseqlist // Last

{1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0,
 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0,
 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0,
 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0,
 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1,
 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0,
 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0}

AutocorrelationSequence[Goldseqlist // Last]

{255, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
 -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1}
```

The third sequence in the list has the following autocorrelation

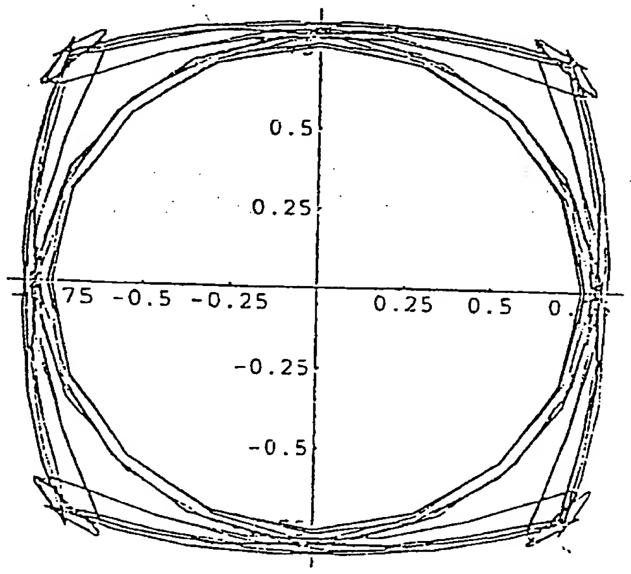
```
AutocorrelationSequence[Goldseqlist // #[[3]]&]

{255, -1, -1, -1, -1, -1, -1, -17, -1, -1, -1, -17, -1, -17, -17, -17, -1, -17, -1, -1,
 -1, -1, 15, -1, -1, 15, 15, -17, -17, -1, -17, 15, -1, -1, -17, 15, -1, 15, -1, 15, -1,
 15, -1, -1, 15, -17, -1, 15, -1, -1, -17, -17, -17, -1, -17, 15, 15, 15, -1, -1, 15,
 -17, 15, -1, -1, -1, -1, 15, 15, -1, 15, -1, -1, -17, -17, 15, -1, -1, 15, 15, -1, -1,
 -17, -1, -1, -17, -1, -1, 15, -1, 15, -17, -1, -1, -17, -1, -1, 15, -1, -17, 15, -1, 15,
 -1, 15, 15, -1, 15, -17, 15, 15, -1, 15, -1, 15, -17, -1, -1, -1, 15, -17, -17, 15, -1,
 -17, -1, -1, -1, -1, -1, -1, -17, -1, 15, -17, -17, 15, -1, -1, -1, -17, 15, -1, 15, -1,
 15, 15, -17, 15, -1, 15, 15, -1, 15, -1, 15, -17, -1, 15, -1, -1, -17, -1, -1, -17,
 15, -1, 15, -1, -1, -17, -1, -1, -17, -1, -1, 15, 15, -1, -1, 15, -17, -17,
 -1, -1, 15, -1, 15, 15, -1, -1, -1, -1, 15, -17, 15, -1, -1, 15, 15, 15, -17,
 -1, -17, -17, -17, -1, -1, 15, -1, -17, 15, -1, -1, 15, -1, 15, -1, 15, -1,
 15, -17, -1, -1, 15, -17, -1, -17, -17, 15, 15, -1, -1, 15, -1, -1, -1, -1,
 -17, -1, -17, -17, -17, -1, -17, -1, -1, -1, -17, -1, -1, -1, -1, -1, -1}
```

We generate the output of the modulator (GMSK modulation)

```
ModOutput = Modulator[L][(Goldseqlist // Last) /. (0 -> -1), NumberOfCurves → 2,
  ModulatingPulse → FiltPulse, SamplingInterval → T/4];
```

We check the output

```
ListPlot[{Re[ModOutput], Im[ModOutput]} // Transpose, PlotJoined -> True,
 AspectRatio -> 1]
```



Filtered pulse

```
- Graphics -
```

Now we try to test the modulated sequence using the

Building a receiver based on pulse length

The number of samples per chip is equal to

$$\frac{3}{26000000} \Big/ T$$

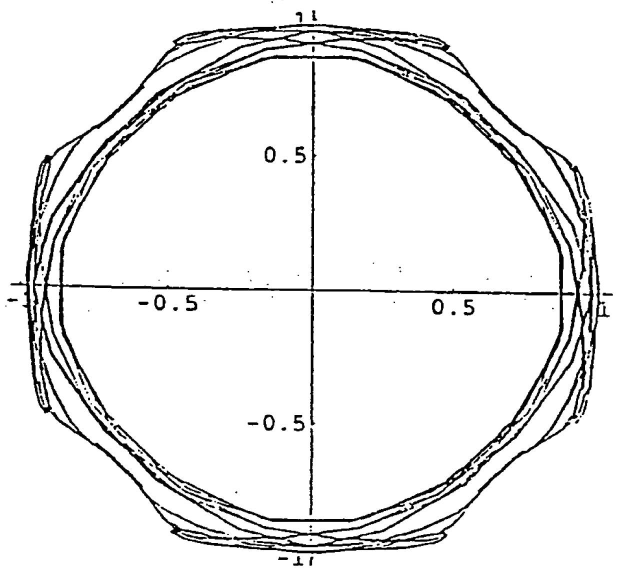
$$\frac{1}{32}$$

A check:- comparison of the received signal with what was being sent.

```
(Receiver[L][ModOutput, StartingQuadrant -> 0,
    SamplingInterval → T / 4, ModulatingPulse → FiltPulse] // Drop[#, 4]& ) -
 ( (Goldseqlist // Last // Drop[#, -4]&) /. ( 0 -> -1))

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

We have successfully demodulated the bitstream

```
ModOutputOpt = Modulator[L][(Goldseqlist // Last) /. (0 -> -1),
  NumberOfCurves → 2, ModulatingPulse → OptPulseScaled, SamplingInterval → T / 4];

ListPlot[{Re[ModOutputOpt], Im[ModOutputOpt]} // Transpose, PlotJoined -> True,
 AspectRatio -> 1]
```



An alternative modulator output

- preferred pulse shape

```
- Graphics -
```

Another check:-

```
(Receiver[L][ModOutputOpt, StartingQuadrant -> 0,
    SamplingInterval → T / 4, ModulatingPulse → OptPulseScaled] // Drop[#, 4]& ) -
 ( (Goldseqlist // Last // Drop[#, -4]&) /. ( 0 -> -1))

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

⊟ CDMA OPERATION

As an example given a symbol stream e.g. {1,1,−1,−1, ...} consisting of −1 and 1, the following function demonstrates the encoding process

Simulation of encoding at transmitter

```
CDMAEncode[BiPolarBitSeq_, GoldSeq_] :=
  Module[{x1, x2, x3},
  x1 = GoldSeq /. {0 → -1};
  Map[x1 # &, BiPolarBitSeq] // Flatten]

CDMAEncodedSeq = CDMAEncode[{-1, 1, 1, -1}, Goldseqlist // Last];
```

CDMA decoding of single Symbol

The modular output associated with {1}

```
ModOutputPlusOne = Modulator[L][CDMAEncode[{1}, (Goldseqlist // Last) /. {0 -> -1}],
  NumberOfCurves → 2, ModulatingPulse → OptPulseScaled, SamplingInterval → T / 4];
```

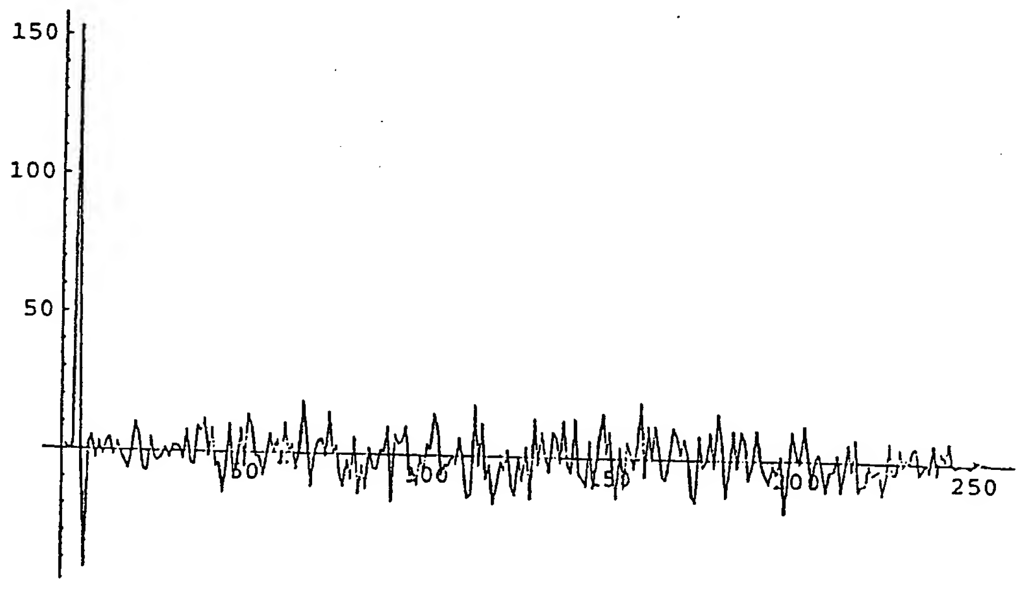
This is a primitive decoder built to study the autocorrelation. This will help in decoding

```
Take[ModOutputPlusOne, 10]

{0.691379 + 0.510132 I, 0.431257 + 0.748432 I, 0.130196 + 0.86360[illegible] I,
 -0.183585 + 0.853405 I, -0.510127 + 0.69136 I, -0.748423 + 0.43[illegible] I,
 -0.863782 + 0.130145 I, -0.853326 - 0.183682 I, -0.69115 - 0.510[illegible] I,
 -0.430757 - 0.74887 I}

PrimitiveCDMAReceiver[ModOutput_, GoldSeq_, Sample_, OverSampling_] :=
 Module[{x1, x2State, x3Update, x4, x5State, x6},
  x1 = Partition[ModOutput, OverSampling] // Transpose // #[[Sample]]&;
  x2State = GoldSeq /. {0 -> -1};
   x5State = Table[(-1)^Mod[i, 2], {i, 0, Length[x2State] - 1}];
  x3Update := Module[{},
    x2State = RotateRight[x2State]; x4 = FoldList[Plus, 0, x2State] // Rest // I^#&;
   x5State = RotateRight[x5State];  x1 (x5State x4) // Apply[Plus, #]&];
Table[x3Update, {i, 1, Length[GoldSeq]}]]
```

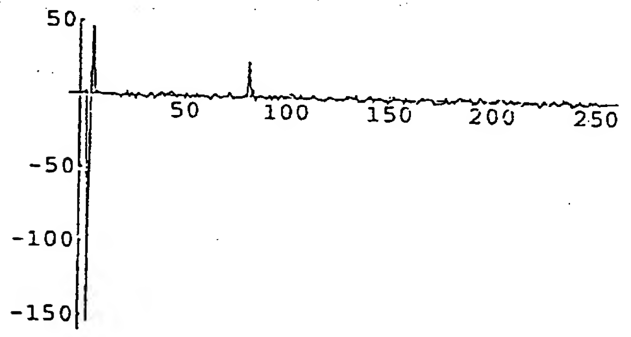
We make it more efficient

```
PrimitiveCDMAReceiver2[ModOutput_, GoldSeq_, Sample_, OverSampling_] :=
 Module[{x1, x2State, x3Update, x4, x5State, x6},
  x1 = Partition[ModOutput, OverSampling] // Transpose // #[[Sample]]&;
  x2State = GoldSeq /. {0 -> -1};
   x5State = Table[(1)^Mod[i, 2], {i, 0, Length[x2State] - 1}];
  x3Update := Module[{},
    x2State = RotateRight[x2State]; x4 = FoldList[Plus, 0, x2State] // Rest // I^-#&;
   x5State = RotateRight[x5State];  x1 (x5State x4) // Apply[Plus, #]&];
Table[x3Update, {i, 1, Length[GoldSeq]}]]

Tom = PrimitiveCDMAReceiver[ModOutputPlusOne, (Goldseqlist // Last), 1, 4];
```

```
ListPlot[Tom // Re, PlotJoined -> True, PlotRange -> All]
```

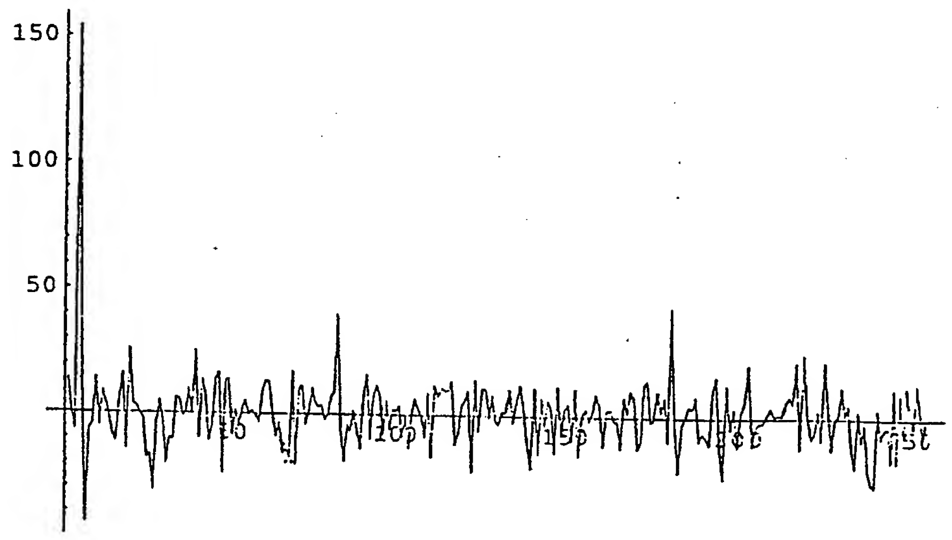


```
- Graphics -

Tom = PrimitiveCDMAReceiver2[ModOutputPlusOne, (Goldseqlist // Last), 1, 4];
ListPlot[Tom // Re, PlotJoined -> True, PlotRange -> All]
```

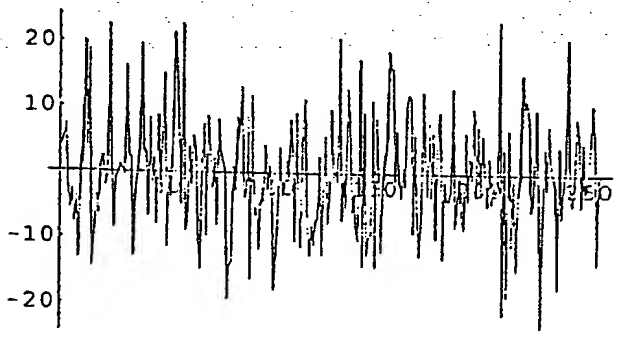


```
- Graphics -

Tom3 = PrimitiveCDMAReceiver2[ModOutputPlusOne3, (Goldseqlist // #[[3]]&), 1, 4];
```

```
ListPlot[Tom3 // Re, PlotJoined -> True, PlotRange -> All]
```

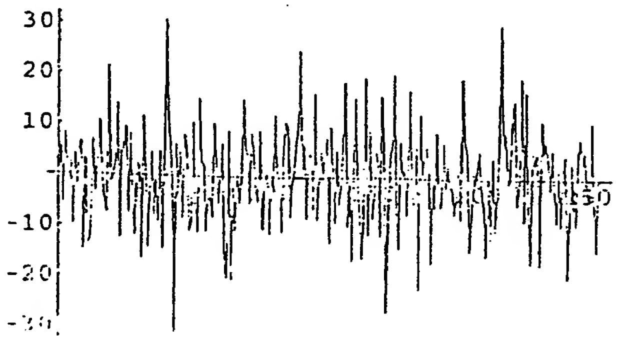


```
- Graphics -

Tom4 = PrimitiveCDMAReceiver2[ModOutputPlusOne3, (Goldseqlist // #[[4]]&), 1, 4];

ListPlot[Tom4 // Re, PlotJoined -> True, PlotRange -> All]
```



'Wrong' Gold Code

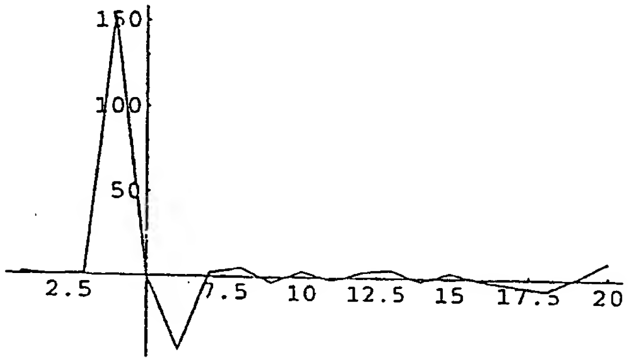
```
- Graphics -

Tom5 = PrimitiveCDMAReceiver[ModOutputPlusOne3, (Goldseqlist // #[[4]]&), 1, 4];

ListPlot[Tom5 // Re, PlotJoined -> True, PlotRange -> All]
```



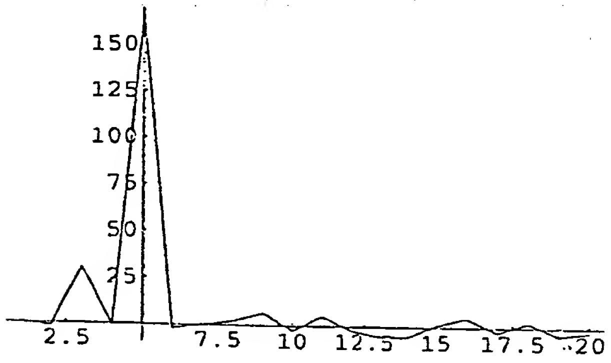
'Wrong' Gold Code

```
- Graphics -
```

```
Length[Tom]

255

Take [Tom, 20]

{1.75501 + 0.329995 I, 0.0343421 - 1.58366 I, 1.39349 + 29.6902 I, 153.017 - 0.957068 I,
 -0.718377 + 165.563 I, -43.6424 - 2.04777 I, 2.21031 + 0.256635 I, 5.392 + 2.18642 I,
 -3.13017 + 6.54167 I, 3.61193 - 2.5452 I, -1.68586 + 5.24864 I, 3.33577 - 1.96193 I,
 4.93409 - 5.10993 I, -1.90672 - 5.29083 I, 3.34843 + 0.914983 I, -1.24192 + 4.93512 I,
 -3.90572 - 2.50768 I, -6.70085 + 2.60649 I, 0.886488 - 4.00636 I, 11.5807 - 2.08322 I}

ListPlot[Tom // Re // Take[#, 20]&, PlotJoined -> True, PlotRange -> All]
```

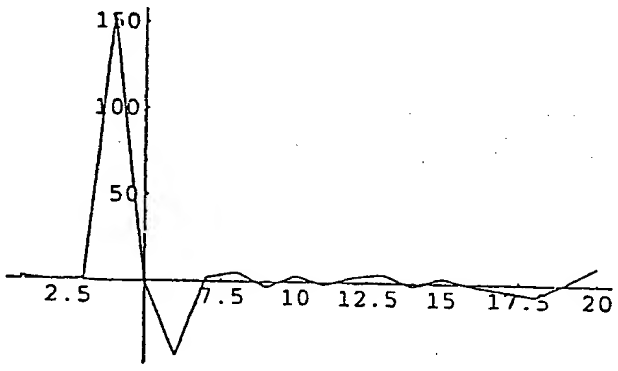


```
- Graphics -

ListPlot[Tom // Im // Take[#, 20]&, PlotJoined -> True, PlotRange -> All]
```



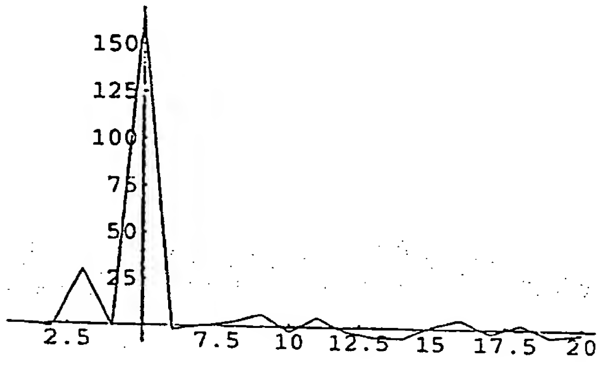
```
- Graphics -

Take[Tom, 10]

{1.75501 + 0.329995 I, 0.0343421 - 1.58366 I, 1.39349 + 29.6902 I, 153.017 - 0.957068 I,
 -0.718377 + 165.563 I, -43.6424 - 2.04777 I, 2.21031 + 0.256635 I, 5.392 + 2.18642 I,
 -3.13017 + 6.54167 I, 3.61193 - 2.5452 I}
```

```
ListPlot[Tom // Re // Take[#, 20]&, PlotJoined -> True, PlotRange -> All]
```



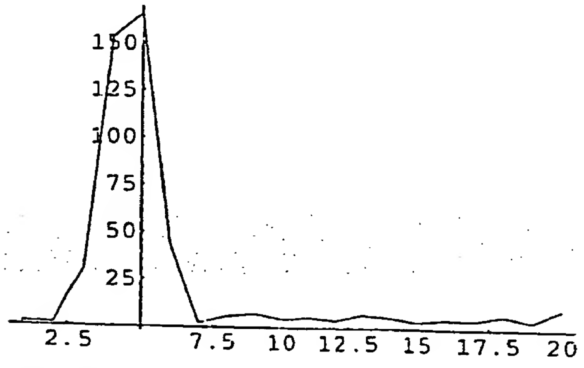
```
- Graphics -
```

```
ListPlot[Tom // Im // Take[#, 20]&, PlotJoined -> True, PlotRange -> All]
```



```
- Graphics -
```

```
ListPlot[Tom // Abs // Take[#, 20]&, PlotJoined -> True, PlotRange -> All]
```

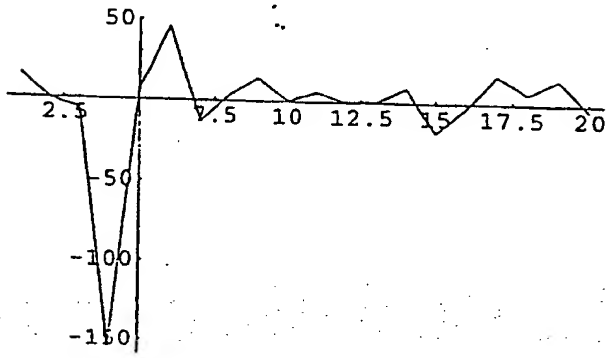


```
- Graphics -
```

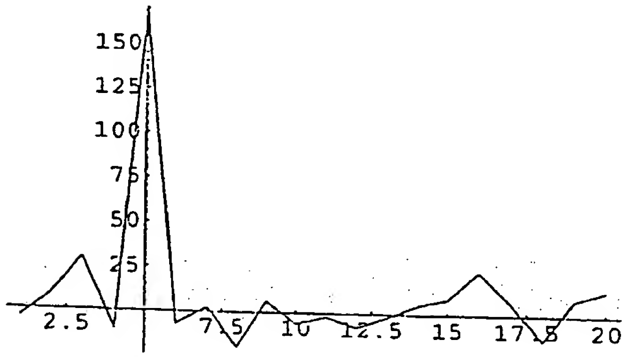
We try a less favourable sequence

```
ModOutputPlusOne3 = Modulator[L][CDMAEncode[{1}, Goldseqlist // #[[3]]&],
  NumberOfCurves -> 2, ModulatingPulse -> OptPulseScaled, SamplingInterval -> T/4];
```

```
PrimitiveCDMAReceiverMinus[ModOutput_, GoldSeq_, Sample_, OverSampling_] :=
 Module[{x1, x2State, x3Update, x4, x5State, x6},
  x1 = Partition[ModOutput, OverSampling] // Transpose // #[[Sample]]&;
  x2State = -(GoldSeq /. {0 -> -1});
   x5State = Table[(-1)^Mod[i, 2], {i, 0, Length[x2State] - 1}];
  x3Update := Module[{},
    x2State = RotateRight[x2State]; x4 = FoldList[Plus, 0, x2State] // Rest // I^#&;
   x5State = RotateRight[x5State]; x1 (x5State x4) // Apply[Plus, #]&];
 Table[x3Update, {i, 1, Length[GoldSeq]}]]

Tom4 = PrimitiveCDMAReceiver[ModOutputPlusOne3, (Goldseqlist // #[[3]]&), 1, 4];

Take[Tom4, 10]

{14.7675 - 4.24542 I, -0.301874 + 8.61802 I, -5.92939 + 29.6038 I, -154.61 - 10.4754 I,
 6.64138 + 166.152 I, 45.2887 - 7.16233 I, -12.809 + 2.47796 I, 3.45948 - 19.7653 I,
 14.2097 + 6.33624 I, 0.585872 - 6.10244 I}

ListPlot[Tom4 // Re // Take[#, 20]&, PlotJoined -> True, PlotRange -> All]
```

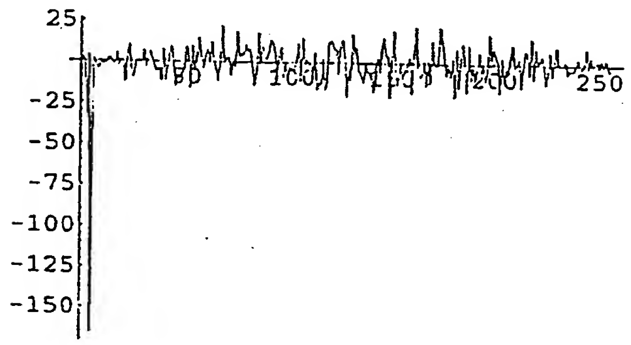


```
- Graphics -

ListPlot[Tom4 // Im // Take[#, 20]&, PlotJoined -> True, PlotRange -> All]
```

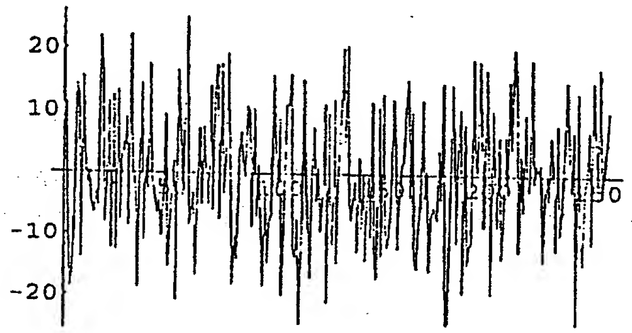


```
- Graphics -

ModOutputMinusOne =
 Modulator[L][CDMAEncode[{-1}, (Goldseqlist // Last) /. {0 -> -1}],
  NumberOfCurves -> 2, ModulatingPulse -> OptPulseScaled, SamplingInterval -> T/4];

TomM1 = PrimitiveCDMAReceiverMinus[ModOutputMinusOne, (Goldseqlist // Last), 1, 4];
```

```
ListPlot[TomM1 // Im, PlotJoined -> True, PlotRange -> All]
```



```
- Graphics -

TomM1 = PrimitiveCDMAReceiver[ModOutputMinusOne, (Goldseqlist // Last), 1, 4];

ListPlot[TomM1 // Im, PlotJoined -> True, PlotRange -> All]
```



Normal receiver for +1 used to try to detect a -1.

```
- Graphics -

Length[Tom]

255

Take [Tom, 20]

{1.75501 + 0.329995 I, 0.0343421 - 1.58366 I, 1.39349 + 29.6902 I, 1[illegible]3.017 - 0.957058 I,
 -0.718377 + 165.563 I, -43.6424 - 2.04777 I, 2.21031 + 0.256636 I, [illegible].392 + 2.18642 I,
 -3.13017 + 6.54167 I, 3.61193 - 2.5452 I, -1.68586 + 5.24864 I, 3.3[illegible]77 - 1.96193 I,
 4.93408 - 5.10993 I, -1.90672 - 5.29083 I, 3.34843 + 0.914983 I, -1.24192 + 4.93512 I,
 -3.90572 - 2.50768 I, -6.70085 + 2.60649 I, 0.886488 - 4.00636 I, 1[illegible].5807 - 2.08322 I}
```

Checking the correlation properties of the Training Sequence in GSM

```
GSM = {0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1}

{0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, [illegible] 1, 1}

GSMseq = {-1, -1, 1, -1, -1, 1, -1, 1, 1, 1, -1, -1, -1, -1, 1, -1, -1, -1, 1,
   -1, -1, 1, -1, 1, 1, 1}

{-1, -1, 1, -1, -1, 1, -1, 1, 1, 1, -1, -1, -1, -1, 1, -1, -1, -1, [illegible] -1, -1,
 1, -1, 1, 1, 1}
```

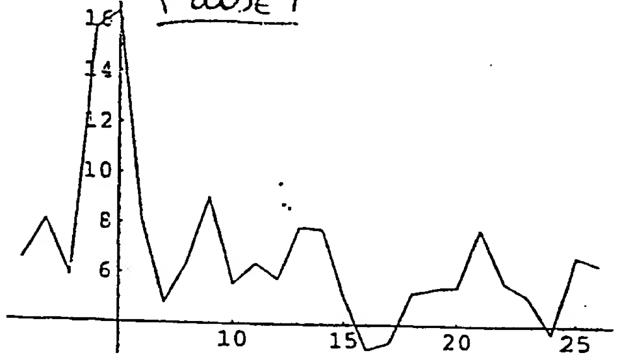
GSMseq = GSM training sequence used as despreading code.

```
AutocorrelationSequence[GSMseq]
```

– Checking correlation properties of GSM train sequence.

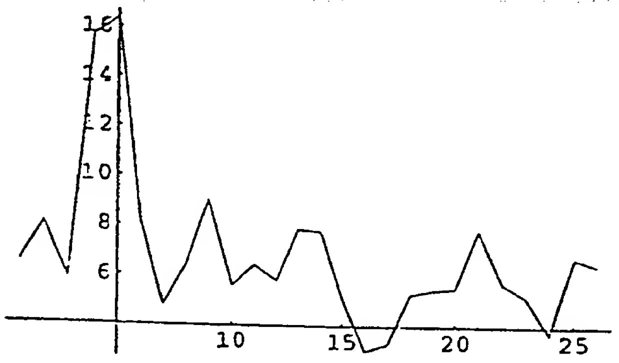
```
{26, -2, -2, 2, -2, -2, -2, 6, -10, 2, 10, -2, -2, -2, -2, -2, 10, 2, -10, 6,
 -2, -2, -2, 2, -2, -2}

ModOutputGSM = Modulator[L][GSMseq, NumberOfCurves → 2,
  ModulatingPulse → OptPulseScaled, SamplingInterval → T / 4];

TomGSM1 = PrimitiveCDMAReceiver[ModOutputGSM, GSMseq, 1, 4];

ListPlot[TomGSM1 // Abs, PlotJoined -> True, PlotRange -> All]
```

Pulse 1



Transformation 1

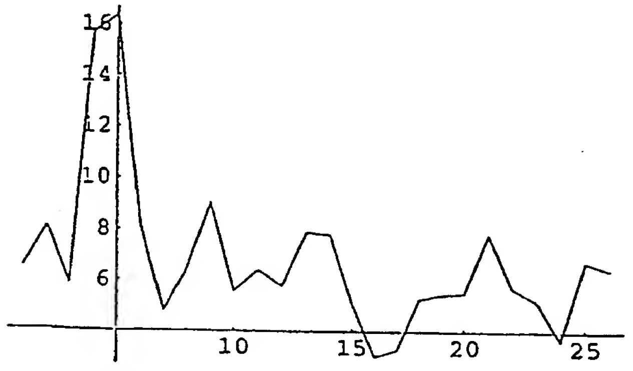
```
- Graphics -

PrimitiveCDMAReceiver2[ModOutputGSM, GSMseq, 1, 4];

ListPlot[%90 // Abs, PlotJoined -> True, PlotRange -> All]
```



Transformation 1b.

```
- Graphics -

PrimitiveCDMAReceiver[ModOutputGSM, GSMseq, 1, 4];
```

```
ListPlot[%92 // Abs, PlotJoined -> True, PlotRange -> All]
```

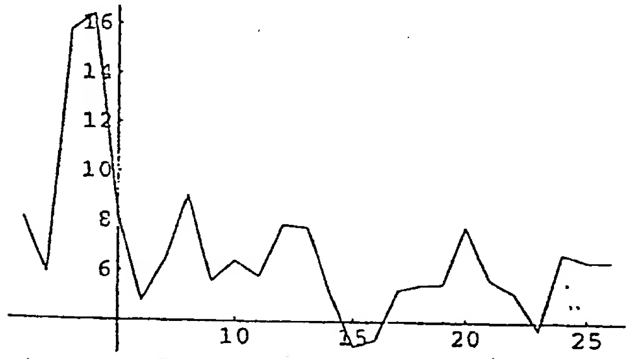


```
- Graphics -
```

```
PrimitiveCDMAReceiverGSM2Pulse[ModOutput_, GoldSeq_, Sample_, OverSampling_] :=
 Module[{x1, x2State, x3Update, x4, x5State, x5, x6},
  x1 =  Partition[ModOutput, OverSampling] // Transpose // #[[Sample]]&;
  x2State = GoldSeq /. {0 -> -1};
   x5State = Table[(-1)^Mod[i, 2], {i, 0, Length[x2State] - 1}];
  x3Update := Module[{}, x2State = RotateRight[x2State];
    x4 = FoldList[Plus, 0, x2State] // Rest;
      x5 = Join[{1}, x2State // Drop[#, -1]&];
      x6 = FoldList[Plus, 0, x5] // Rest // I^#&;
   x5State = RotateRight[x5State];  x1 (x5State x6) // Apply[Plus, #]&];
Table[x3Update, {i, 1, Length[GoldSeq]}]]

TomGSMSecondP = PrimitiveCDMAReceiverGSM2Pulse[ModOutputGSM, GSMseq, 1, 4];
```

Pulse 2

```
ListPlot[TomGSMSecondP // Abs, PlotJoined -> True, PlotRange -> All]
```

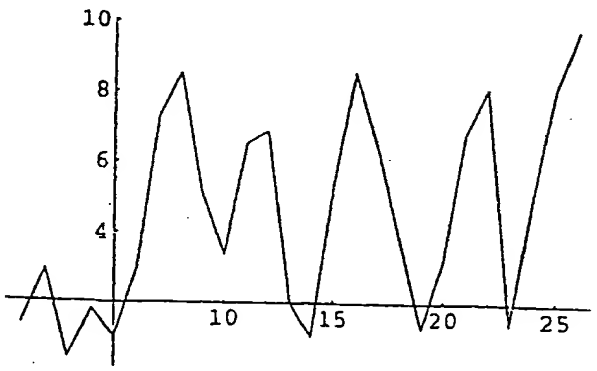


Transformation 3

```
- Graphics -
```

```
PrimitiveCDMAReceiverGSM2PulseEfficient[
  ModOutput_, GoldSeq_, Sample_, OverSampling_] :=
 Module[{x1, x2State, x3Update, x4, x5State, x5, x6},
  x1 =  Partition[ModOutput, OverSampling] // Transpose // #[[Sample]]&;
  x2State = GoldSeq /. {0 -> -1};
   x5State = Table[(1)^Mod[i, 2], {i, 0, Length[x2State] - 1}];
  x3Update := Module[{}, x2State = RotateRight[x2State];
    x4 = FoldList[Plus, 0, x2State] // Rest;
      x5 = Join[{1}, x2State // Drop[#, -1]&];
      x6 = FoldList[Plus, 0, x5] // Rest // I^#&;
   x5State = RotateRight[x5State];  x1 (x5State x6) // Apply[Plus, #]&];
Table[x3Update, {i, 1, Length[GoldSeq]}]]

TomGSMSecondPEff2 =
 PrimitiveCDMAReceiverGSM2PulseEfficient[ModOutputGSM, GSMseq, 1, 4];
```

```
ListPlot[TomGSMSecondPEff2 // Abs, PlotJoined -> True, PlotRange -> All]
```



- Graphics -

Transformation 3b

▫ CDMA Decoding of Several Symbols with Training sequence Receiver

First we generate the modulator output. For simplicity we will use a very short training sequence . Let the training sequence one of GSM training sequences Let the guard sequences be {1,1,1} . Let the data symbols be generated by a random

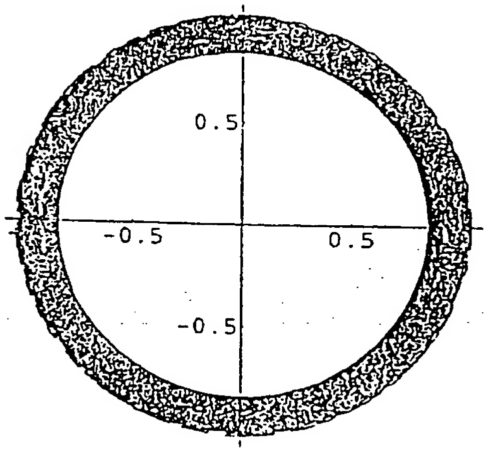
```
data1 = {1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0};

data2 = {0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1};

guard = {1, 1, 1}

{1, 1, 1}

training = {0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1};
```

The GSM Training sequence is {0,0,1,0,0,1,0,1,1,1,0,0,0,0,1,0,0,0,1,0,0,1,0,,1,1,1}. In fact any short m−sequence can be used to characterise the output.

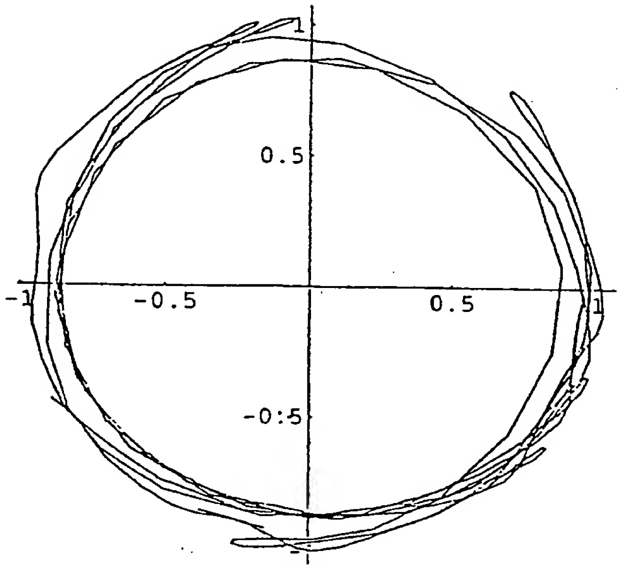
Only at this point that we differentially encode using the gsm scheme

```
frame = Join[guard, data1, training, data2, guard]

General::spell1 :
 Possible spelling error: new symbol name "frame" is similar to existing symbol "Frame".

{1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0,
 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1,
 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1}

GSMDiffEncodedFrame[frame_] := Module[{x1, x2, x3}, x1 = Partition[frame, 2, 1];
  x2 = Map[Mod[#[[1]] + #[[2]], 2]&, x1] // Join[{frame[[1]]}, #]&;
 x3 = x2 /. {0 -> 1, 1 -> -1}]

General::spell1 :
 Possible spelling error: new symbol name "frame" is similar to existing symbol "Frame".

frameEncoded = GSMDiffEncodedFrame[frame]

{-1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, 1, 1, -1, -1, 1, -1, -1, -1, 1, 1,
 -1, -1, 1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, -1, 1, 1, -1, -1, 1, -1, -1, -1,
 1, 1, -1, 1, -1, -1, -1, 1, -1, -1, -1, -1, 1, -1, -1, 1, -1, -1, -1, 1, -1, 1, 1, 1, 1}

ModOutputFrame3 = Modulator[1][CDMAEncode[frameEncoded, Column[illegible] // Last],
  NumberOfCurves -> 2, ModulatingPulse -> OptPulseScaled, SamplingInterval -> T/4];
```

```
Carrfreq001 = Table[E^(I 2 Pi 0.001 j) // N, {j, 0, Length[ModOutputFrame3] - 1}];

Carr001 = ModOutputFrame3 Carrfreq001;

Save["Carr001.m", Carr001];

Carrfreq0005 = Table[E^(I 2 Pi 0.0005 j) // N, {j, 0, Length[ModOutputFrame3] - 1}];

Carr0005 = ModOutputFrame3 Carrfreq0005;

Save["Carr0005.m", Carr0005];

Carrfreq002 = Table[E^(I 2 Pi 0.002 j) // N, {j, 0, Length[ModOutputFrame3] - 1}];

Carr002 = ModOutputFrame3 Carrfreq002;

Save["Carr002.m", Carr002];

ModOutputUnEncodedFrame3 =
 Modulator[L][ CDMAEncode[frame /. 0 -> -1, Goldseqlist // #[[3]]&],
  NumberOfCurves → 2, ModulatingPulse → OptPulseScaled, SamplingInterval → T/4];

Save["ModOutputUnEncodedFrameGSMLike3.m", ModOutputFrame]

<< ModOutputFrameEncodedGSMLike.m;

Length[ModOutputFrame]

73440

AFC0005 = Take[Carr0005, {50, 7300}];

AFC0005 // {Re[#], Im[#]}& // Transpose //
 ListPlot[#, PlotJoined -> True, PlotRange -> All, AspectRatio -> 1]&;
```



```
AFC0005 // Take[#, 200]& // {Re[#], Im[#]}& // Transpose //
 ListPlot[#, PlotJoined -> True, PlotRange -> All, AspectRatio -> 1]&;
```



In the PrimitiveCDMA receiver we need to specify the sample. In the CDMA synchroniser we discover the sample.

[illegible]

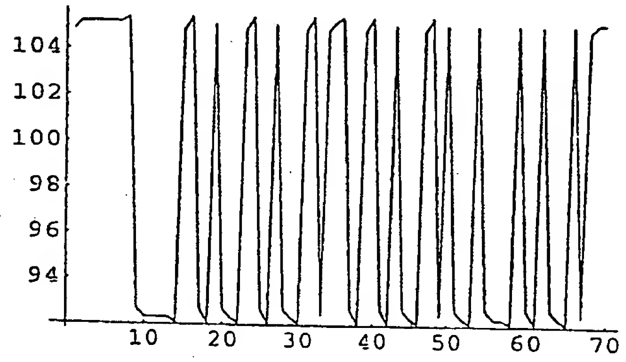
```
CDMACoarseSynchroniserNew[ModOutput_, GoldSeq_, Threshold_, OverSampling_] :=
 Module[{x1, x2, x3, x4Plus, x4Minus, x5State,
   x6Count, x6MaxCorr, seq, x7Update, x8, x9, x10, x11, x12, x13},
  x1 = ModOutput;
  x2 = GoldSeq /. 0 -> -1;
  x3 = CodingTransformNew[L][GoldSeq];
  x4Plus = x3[[1]];
  x4Minus = x3[[2]];
  x5State = Take[x1, (Length[x2] + 1) OverSampling];
  x6Count = 1;
  x6MaxCorr = 0;
  seq = Drop[x1, OverSampling (Length[x2] + 1)];
 x7Update := Module[{},
x8 = Partition[x5State, OverSampling] // Transpose;
x9 = Map[{Drop[#, -1] . x4Plus, Drop[#, 1] . x4Minus}&, x8];
x10 = Map[
      {Abs[Im[#[[1]]]], Abs[Re[#[[1]]]], Abs[Re[#[[2]]]], Abs[Im[#[[2]]]] }&, x9];
x11 = If[Max[x10] > Threshold, Throw[{x10, x6Count, True}],
      {x6Count, x10, x6MaxCorr = Max[x6MaxCorr, x10], False}];
    x6Count = x6Count + 1;
    x5State = Join[Drop[x5State, OverSampling], Take[seq, OverSampling]];
    seq = Drop[seq, OverSampling];
    x11 ];
x12 = Catch[Table[x7Update, {i, 1, Length[seq] / OverSampling }]];
 x13 = If[Last[x12] === True,
    CDMAFineSynchroniser[ModOutput, x12[[2]], x3, OverSampling],
    {"Failed to Coarse Synchronise", False}]]

Tom4 =
 CDMACoarseSynchroniserNew[AFC0005 // Drop[#, 250 4]&, Goldseqlist // Last , 50, 4]

DeModulator[
 {{{16.252 - 103.911 I, 9.19889 + 6.93038 I, -3.61972 + 15.2763 I, 13.0046 - 9.41072 I},
   {-102.686 - 22.7446 I, 7.49431 - 8.74558 I, 15.0189 + 4.57178 I, -8.57558 - 13.5698 I},
   {-29.1474 + 101.055 I, -8.25775 - 8.02866 I, 5.50581 - 14.7022 I, -14.0815 + 7.70661 I},
   {99.0255 + 35.4352 I, -8.53133 + 7.73733 I, -14.3275 - 6.4181 I, 6.80721 + 14.5376 I},
   {41.5832 - 96.6051 I, 7.18638 + 9.00033 I, -7.30507 + 13.8962 I, 14.9364 - 5.88096 I}},
  {{24.6533 - 96.8492 I, 10.1266 + 6.85283 I, 1.46802 - 2.37691 I, 12.5087 - 10.488 I},
   {-95.1101 - 30.6858 I, 7.47516 - 9.6763 I,
    -2.28004 - 1.61437 I, -9.68188 - 13.1425 I}, {-36.5973 + 92.9536 I,
    -9.18784 - 8.06795 I, -1.75435 + 2.37418 I, -13.7265 + 8.83755 I},
   {90.5141 + 42.3633 I, -8.62898 + 8.66312 I, 2.05973 + 1.8274 I, 7.95834 + 14.2523 I},
   {47.9641 - 87.6755 I, 8.1042 + 9.15591 I, 2.01301 - 1.93715 I, 14.7239 - 7.04772 I}}}
```

```
Tom0005 =
 CDMACoarseSynchroniserNew[Carr0005 // Drop[#, 250 4]&, Goldseqlist // Last, 100, 4];

CDMAFineSynchroniser[ModOutput_,
  ThresholdCorrelatioCount_, CorrelatingSeq_, OverSampling_] :=
   Module[{x1, x2, x3, x4, x5, x6, seq, x7Update, x8First,
   x8Second, x9First, x9Second, x10First, x10Second, x11, x12, x13, x14},
  x1 = If[ThresholdCorrelatioCount > 3,
    ThresholdCorrelatioCount - 3, ThresholdCorrelatioCount];
  x2 = CorrelatingSeq;
  x3 = Drop[ModOutput, x1 OverSampling];
 x4 =  CDMAPositionFinder[x3, x2, OverSampling];
  x5 = CDMAPositionFinder[Drop[x3, Length[x2] OverSampling] , x2, OverSampling];
   x6 = CDMAPositionFinder[Drop[x3, 2 Length[x2] OverSampling] , x2, OverSampling];
  x7 = PositionAverager[{x4[[1]], x5[[1]], x6[[1]]}];
  x8First = Drop[x3,  (x7[[1]] - 1) OverSampling] //
      Partition[#, OverSampling]& // Transpose // #[[x7[[2]]]]&;
   x8Second = Drop[x3, x7[[1]] OverSampling] //
      Partition[#, OverSampling]& // Transpose // #[[x7[[2]]]]&;
 x9First = x8First // Partition[#, Length[x2[[1]]]]&;
  x10First = Map[Function[x, Map[x . #&, x2]], x9First];
    x9Second = x8Second // Partition[#, Length[x2[[1]]]]&;
  x10Second = Map[Function[x, Map[x . #&, x2]], x9Second] ;
  DeModulator[{x10First, x10Second}]   ]

CDMAPositionFinder[ModOutput_, CorrelatingSeq_, OverSampling_] :=
  Module[{x5State, x6Count, seq, x7Update, x8, x9, x10, x11, x12, x13},
 x5State = Take[ModOutput, (Length[CorrelatingSeq[[1]]] + 1) OverSampling];
  x6Count = 1;
  seq = Drop[ModOutput, OverSampling (Length[CorrelatingSeq[[1]]] + 1)];
  x7Update := Module[{},
x8  = Partition[x5State, OverSampling] // Transpose;
x9  =  Map[(Drop[#, -1] . CorrelatingSeq[[1]], Drop[#, -1] . CorrelatingSeq[[2]],
        Drop[#, 1] . CorrelatingSeq[[1]], Drop[#, 1] . CorrelatingSeq[[2]]}&, x8];
    x6Count = x6Count + 1;
    x5State = Join[Drop[x5State, OverSampling], Take[seq , OverSampling]];
    seq = Drop[seq, OverSampling];
    x9 ];
x10 =  Table[x7Update, {i, 1, 10}];
x11 =  MapIndexed[Max[{Abs[Re[#]], Abs[Im[#]]}]&, x10, {3}];
x12 = MapIndexed[Apply[Plus, #]&, x11, {2}];
  x13 = Position[x12, Max[x12]] ]
```

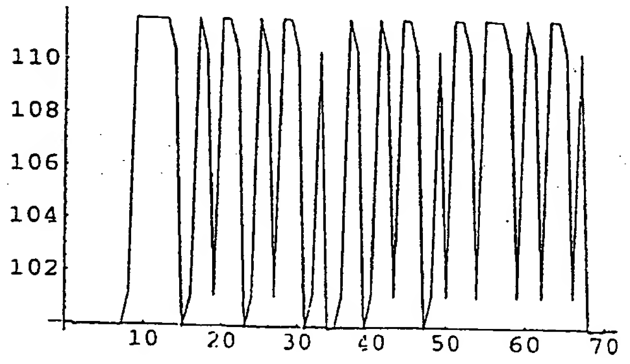
We need to define a position averager. We for now just take the first element

```
PositionAverager[PositionList_] := First[PositionList]

Tom4 =
 CDMACoarseSynchroniserNew[TestData // Drop[#, 250 4]&, Goldseqlist // Last, 100, 4]

DeModulator[
 {{{-157.056 + 0.691379 I, 7.37139 - 18.8305 I, 41.1322 + 1.94693 I, -16.2843 - 23.4715 I},
   {-0.691379 - 157.056 I, 18.8305 + 7.37139 I, -1.94693 + 41.1322 I, 23.4715 - 16.2843 I},
   {157.056 - 0.691379 I, -7.37139 + 18.8305 I, -41.1322 - 1.94693 I, 16.2843 + 23.4715 I},
   {0.691379 + 157.056 I, -18.8305 - 7.37139 I,
    1.94693 - 41.1322 I, -23.4715 + 16.2843 I}, {-157.056 + 0.691379 I,
    7.37139 - 18.8305 I, 41.1322 + 1.94693 I, -16.2843 - 23.4715 I}},
  {{-169.734 - 0.510132 I, 6.8871 - 20.5026 I, 6.24607 + 1.47947 I, -17.4944 - 21.5101 I},
   {0.510132 - 169.734 I, 20.5026 + 6.8871 I, -1.47947 + 6.24607 I, 21.5101 - 17.4944 I},
   {169.734 + 0.510132 I, -6.8871 + 20.5026 I, -6.24607 - 1.47947 I, 17.4944 + 21.5101 I},
   {-0.510132 + 169.734 I, -20.5026 - 6.8871 I, 1.47947 - 6.24607 I, -21.5101 + 17.4944 I},
   {-169.734 - 0.510132 I, 6.8871 - 20.5026 I, 6.24607 + 1.47947 I,
    -17.4944 - 21.5101 I}}}]
```

```
Tom4[[1]] // Transpose

{{{-157.056 + 0.691379 I, 7.37139 - 18.8305 I, 41.1322 + 1.94693 I, -16.2843 - 23.4715 I},
  {-169.734 - 0.510132 I, 6.8871 - 20.5026 I, 6.24607 + 1.47947 I, -17.4944 - 21.5101 I}},
 {{-0.691379 - 157.056 I, 18.8305 + 7.37139 I, -1.94693 + 41.1322 I, 23.4715 - 16.2843 I},
  {0.510132 - 169.734 I, 20.5026 + 6.8871 I, -1.47947 + 6.24607 I, 21.5101 - 17.4944 I}},
 {{157.056 - 0.691379 I, -7.37139 + 18.8305 I, -41.1322 - 1.94693 I, 16.2843 + 23.4715 I},
  {169.734 + 0.510132 I, -6.8871 + 20.5026 I, -6.24607 - 1.47947 I, 17.4944 + 21.5101 I}},
 {{0.691379 + 157.056 I, -18.8305 - 7.37139 I, 1.94693 - 41.1322 I, -23.4715 + 16.2843 I},
  {-0.510132 + 169.734 I, -20.5026 - 6.8871 I, 1.47947 - 6.24607 I, -21.5101 + 17.4944 I}},
 {{-157.056 + 0.691379 I, 7.37139 - 18.8305 I, 41.1322 + 1.94693 I, -16.2843 - 23.4715 I},
  {-169.734 - 0.510132 I, 6.8871 - 20.5026 I, 6.24607 + 1.47947 I, -17.4944 - 21.5101 I}}}

Take[Tom5[[1]] // Transpose, {8, 10}]

{{{157.25 - 0.929033 I, -7.57128 + 19.0518 I, -40.9109 - 1.74704 I, 16.522 + 23.6653 I},
  {169.813 - 0.714545 I, -7.42576 + 21.3328 I, -5.4158 - 0.940778 I, 18.7191 + 21.5891 I}},
 {{19.1272 - 6.93509 I, -1.02745 + 156.682 I, 23.9098 + 15.9863 I, -1.57069 - 41.4696 I},
  {20.5326 - 6.8667 I, 0.478141 + 169.715 I, 21.5305 + 17.4644 I, -1.46045 - 6.27806 I}},
 {{-7.37139 - 18.8305 I, 157.056 + 0.691379 I, 16.2843 - 23.4715 I, -41.1322 + 1.94693 I},
  {-6.8871 - 20.5026 I, 169.734 - 0.510132 I, 17.4944 - 21.5101 I, -6.24607 + 1.47947 I}}}

Tom6[[1, 1]] // Transpose // {#[[1]], #[[2]]}& // Transpose // Map[Max, Abs[#]]& //
 ListPlot[#, PlotJoined -> True]&
```



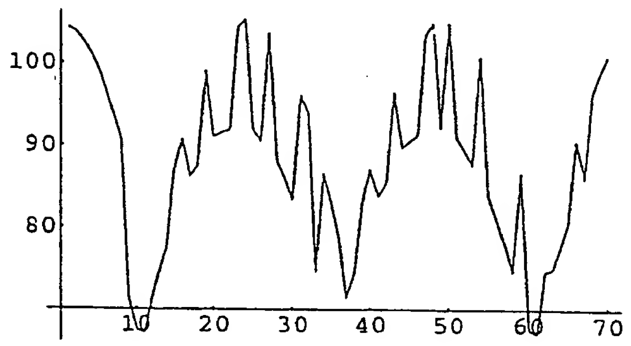
```
- Graphics -

Tom6[[1, 2]] // Transpose // {#[[1]], #[[2]]}& // Transpose // Map[Max, Abs[#]]& //
 ListPlot[#, PlotJoined -> True]&
```



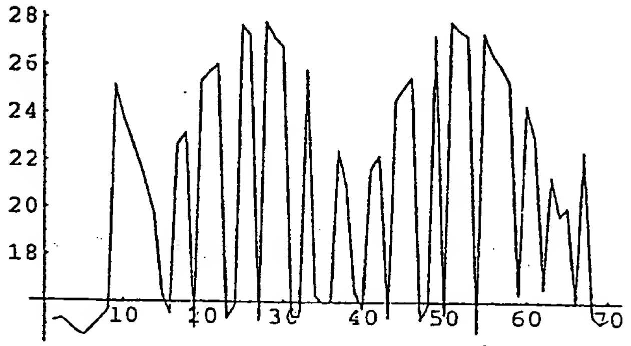
```
- Graphics -
```

```
Map[Abs, Tom6[[1, 2]]]

{{99.9069, 12.1968, 2.82297, 16.3585}, {99.9377, 12.2274, 2.7937, 16.3237},
 {99.9377, 12.2274, 2.7937, 16.3237}, {99.9377, 12.2274, 2.7937, 16.3237},
 {99.9377, 12.2274, 2.7937, 16.3237}, {99.9377, 12.2274, 2.7937, 16.3237},
 {99.9377, 12.2274, 2.7937, 16.3237}, {101.16, 12.0496, 2.71211, 17.0171},
 {10.9504, 111.653, 17.2325, 16.2024}, {10.9205, 111.621, 17.2666, 16.1692},
 {10.9205, 111.621, 17.2666, 16.1692}, {10.9205, 111.621, 17.2666, 16.1692},
 {10.9205, 111.621, 17.2666, 16.1692}, {11.1111, 110.39, 16.592, 16.2327},
 {99.9069, 12.1968, 2.82297, 16.3585}, {101.16, 12.0496, 2.71211, 17.0171},
 {10.9504, 111.653, 17.2325, 16.2024}, {11.1111, 110.39, 16.592, 16.2327},
 {101.129, 12.0176, 2.74759, 17.0524}, {10.9504, 111.653, 17.2325, 16.2024},
 {10.9205, 111.621, 17.2666, 16.1692}, {11.1111, 110.39, 16.592, 16.2327},
 {99.9069, 12.1968, 2.82297, 16.3585}, {101.16, 12.0496, 2.71211, 17.0171},
 {10.9504, 111.653, 17.2325, 16.2024}, {11.1111, 110.39, 16.592, 16.2327},
 {101.129, 12.0176, 2.74759, 17.0524}, {10.9504, 111.653, 17.2325, 16.2024},
 {10.9205, 111.621, 17.2666, 16.1692}, {11.1111, 110.39, 16.592, 16.2327},
 {99.9069, 12.1968, 2.82297, 16.3585}, {101.16, 12.0496, 2.71211, 17.0171},
 {11.1392, 110.422, 16.5587, 16.2668}, {99.9069, 12.1968, 2.82297, 16.3585},
 {99.9377, 12.2274, 2.7937, 16.3237}, {101.16, 12.0496, 2.71211, 17.0171},
 {10.9504, 111.653, 17.2325, 16.2024}, {11.1111, 110.39, 16.592, 16.2327},
 {99.9069, 12.1968, 2.82297, 16.3585}, {101.16, 12.0496, 2.71211, 17.0171},
 {10.9504, 111.653, 17.2325, 16.2024}, {11.1111, 110.39, 16.592, 16.2327},
 {101.129, 12.0176, 2.74759, 17.0524}, {10.9504, 111.653, 17.2325, 16.2024},
 {10.9205, 111.621, 17.2666, 16.1692}, {11.1111, 110.39, 16.592, 16.2327},
 {99.9069, 12.1968, 2.82297, 16.3585}, {101.16, 12.0496, 2.71211, 17.0171},
 {11.1392, 110.422, 16.5587, 16.2668}, {101.129, 12.0176, 2.74759, 17.0524},
 {10.9504, 111.653, 17.2325, 16.2024}, {10.9205, 111.621, 17.2666, 16.1692},
 {11.1111, 110.39, 16.592, 16.2327}, {101.129, 12.0176, 2.74759, 17.0524},
 {10.9504, 111.653, 17.2325, 16.2024}, {10.9205, 111.621, 17.2666, 16.1692},
 {10.9205, 111.621, 17.2666, 16.1692}, {11.1111, 110.39, 16.592, 16.2327},
 {101.129, 12.0176, 2.74759, 17.0524}, {10.9504, 111.653, 17.2325, 16.2024},
 {11.1111, 110.39, 16.592, 16.2327}, {101.129, 12.0176, 2.74759, 17.0524},
 {10.9504, 111.653, 17.2325, 16.2024}, {10.9205, 111.621, 17.2666, 16.1692},
 {11.1111, 110.39, 16.592, 16.2327}, {101.129, 12.0176, 2.74759, 17.0524},
 {11.1392, 110.422, 16.5587, 16.2668}, {99.9069, 12.1968, 2.82297, 16.3585},
 {99.9377, 12.2274, 2.7937, 16.3237}, {99.9377, 12.2274, 2.7937, 16.3237}}

Max0005 =
 Map[{Max[{Re[#[[1]]] // Abs, Im[#[[1]]] // Abs, Re[#[[2]]] // Abs, Im[#[[2]]] // Abs}],
     Max[{Re[#[[3]]] // Abs, Im[#[[3]]] // Abs, Re[#[[4]]] // Abs, Im[#[[4]]] // Abs}]}&,
  Tom6[[1, 1]]]

{{104.351, 15.1379}, {103.911, 15.2763}, {102.686, 15.0189},
 {101.055, 14.7022}, {99.0255, 14.5376}, {95.6051, 14.9364},
 {93.8035, 15.2762}, {90.7175, 15.6415}, {71.4359, 25.1654}, {67.5508, 23.8519},
 {67.0266, 22.9468}, {70.8801, 21.9511}, {74.4539, 20.8688}, {77.6444, 19.7766},
 {86.7967, 16.2927}, {90.7238, 15.4739}, {86.1537, 22.7114}, {87.5152, 23.2037},
 {98.9714, 14.9013}, {91.0278, 25.3506}, {91.5367, 25.7331}, {91.8871, 26.0709},
 {104.382, 15.2963}, {105.346, 15.8922}, {91.9518, 27.7219}, {90.4855, 27.2999},
 {103.704, 15.2097}, {88.1543, 27.846}, {85.9746, 27.2075}, {83.3983, 26.8136},
 {96.1447, 15.3897}, {93.9076, 15.3803}, {74.6965, 25.8353}, {85.612, 16.2707},
 {83.2288, 15.9297}, {79.0562, 16.0518}, {71.3823, 22.4556}, {74.383, 20.923},
 {82.9341, 16.4519}, {87.1763, 15.7147}, {83.839, 21.6473}, {85.5445, 22.25},
 {96.5637, 15.3294}, {89.7552, 24.5667}, {90.6053, 25.0753}, {91.3309, 25.5023},
 {103.567, 15.2284}, {104.931, 15.891}, {92.2522, 27.2697}, {104.992, 15.4722},
 {91.0436, 27.8733}, {89.5104, 27.4849}, {87.6329, 27.2718}, {100.78, 14.7073},
 {83.8748, 27.3795}, {81.0829, 26.501}, {78.152, 25.9901}, {74.6066, 25.3375},
 {86.6792, 16.3378}, {67.5876, 24.357}, {65.9937, 22.9638}, {74.5498, 16.5852},
 {74.9528, 21.37}, {77.7339, 19.7041}, {80.5993, 20.0831}, {90.616, 16.0026},
 {86.0108, 22.4409}, {96.2752, 15.4329}, {93.8878, 15.1306}, {100.942, 15.3636}}
```

```
ListPlot[Max0005 // Transpose // #[[1]]&, PlotJoined -> True]
```



```
- Graphics -

ListPlot[Max0005 // Transpose // #[[2]]&, PlotJoined -> True]
```



```
- Graphics -

Map[{Max[{Re[#[[1]]] // Abs, Im[#[[1]]] // Abs, Re[#[[2]]] // Abs, Im[#[[2]]] // Abs}],
    Max[{Re[#[[3]]] // Abs, Im[#[[3]]] // Abs, Re[#[[4]]] // Abs, Im[#[[4]]] // Abs}]}&,
 Tom5[[1, 2]]]

{{169.715, 21.5305}, {169.734, 21.5101}, {169.734, 21.5101},
 {169.734, 21.5101}, {169.734, 21.5101}, {169.734, 21.5101},
 {169.734, 21.5101}, {169.813, 21.5891}, {169.715, 21.5305}, {169.734, 21.5101},
 {169.734, 21.5101}, {169.734, 21.5101}, {169.734, 21.5101}, {169.813, 21.5891},
 {169.715, 21.5305}, {169.813, 21.5891}, {169.715, 21.5305}, {169.813, 21.5891},
 {169.794, 21.6095}, {169.715, 21.5305}, {169.734, 21.5101}, {169.813, 21.5891},
 {169.715, 21.5305}, {169.813, 21.5891}, {169.715, 21.5305}, {169.813, 21.5891},
 {169.794, 21.6095}, {169.715, 21.5305}, {169.734, 21.5101}, {169.813, 21.5891},
 {169.715, 21.5305}, {169.813, 21.5891}, {169.794, 21.6095}, {169.715, 21.5305},
 {169.734, 21.5101}, {169.813, 21.5891}, {169.715, 21.5305}, {169.813, 21.5891},
 {169.715, 21.5305}, {169.813, 21.5891}, {169.715, 21.5305}, {169.813, 21.5891},
 {169.794, 21.6095}, {169.715, 21.5305}, {169.734, 21.5101}, {169.813, 21.5891},
 {169.715, 21.5305}, {169.813, 21.5891}, {169.794, 21.6095}, {169.794, 21.6095},
 {169.715, 21.5305}, {169.734, 21.5101}, {169.813, 21.5891}, {169.794, 21.6095},
 {169.715, 21.5305}, {169.734, 21.5101}, {169.734, 21.5101}, {169.813, 21.5891},
 {169.794, 21.6095}, {169.715, 21.5305}, {169.813, 21.5891}, {169.794, 21.6095},
 {169.715, 21.5305}, {169.734, 21.5101}, {169.813, 21.5891}, {169.794, 21.6095},
 {169.794, 21.6095}, {169.715, 21.5305}, {169.734, 21.5101}, {169.734, 21.5101}}

demodframe = Sign[Re[ Tom5[[1, 1]] SeqI]] /. {-1 -> 1, 1 -> 0}

{1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0,
 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1,
 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1}
```

The first and last bits have been lost in the processing

```
frame

{1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0,
 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1,
 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1}

Length[frame]

72

truncatedframe = frame // Drop[#, 1]& // Drop[#, -1]&

{1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0,
 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1,
 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1}

demodframe - truncatedframe

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}

Tom6 = CDMACoarseSynchroniser[ModOutputUnEncodedFrame3 // Drop[#, 250 4]&,
  Goldseqlist // #[[3]]& , 100, 4]

$Aborted

Tom6[[1, 1]] SeqI // Re

{-157.37, -157.056, -157.056, -157.056, -157.056, -157.056, -157.056, -156.862,
 9.75774, -9.3815, 9.3815, -9.3815, 9.3815, -9.18122, -157.37, -156.862, 9.75774,
 -9.18122, -157.176, 9.75774, -9.3815, 9.18122, 157.37, 156.862, -9.75774, 9.18122,
 157.176, -9.75774, 9.3815, -9.18122, -157.37, -156.862, 9.55746, 157.37, 157.056,
 156.862, -9.75774, 9.18122, 157.37, 156.862, -9.75774, 9.18122, 157.176, -9.75774,
 9.3815, -9.18122, -157.37, -156.862, 9.55746, 157.176, -9.75774, 9.3815, -9.18122,
 -157.176, 9.75774, -9.3815, 9.3815, -9.18122, -157.176, 9.75774, -9.18122,
 -157.176, 9.75774, -9.3815, 9.18122, 157.176, -9.55746, -157.37, -157.056, -157.056}

Tom6[[1, 1]] SeqI // Im

{-10.0454, -10.3815, -10.3815, -10.3815, -10.3815, -10.3815, -10.3815, -10.62,
 -15.3789, 15.6783, -15.6783, 15.6783, -15.6783, 15.9004, -10.0454, -10.62, -15.3789,
 15.9004, -10.2839, -15.3789, 15.6783, -15.9004, 10.0454, 10.62, 15.3789, -15.9004,
 10.2839, 15.3789, -15.6783, 15.9004, -10.0454, -10.62, -15.601, 10.0454, 10.3815,
 10.62, 15.3789, -15.9004, 10.0454, 10.62, 15.3789, -15.9004, 10.2839,
 15.3789, -15.6783, 15.9004, -10.0454, -10.62, -15.601, 10.2839, 15.3789,
 -15.6783, 15.9004, -10.2839, -15.3789, 15.6783, -15.6783, 15.9004, -10.2839,
 -15.3789, 15.9004, -10.2839, -15.3789, 15.6783, -15.9004, 10.2839, 15.601,
 -10.0454, -10.3815, -10.3815}

Tom6[[1, 2]] SeqI // Re

{-169.751, -169.733, -169.733, -169.733, -169.733, -169.733, -169.733, -169.638,
 10.444, -10.425, 10.425, -10.425, 10.425, -9.86944, -169.751, -169.638, 10.444,
 -9.86944, -169.656, 10.444, -10.425, 9.86944, 169.751, 169.638, -10.444, 9.86944,
 169.656, -10.444, 10.425, -9.86944, -169.751, -169.638, 9.88847, 169.751, 169.733,
 169.638, -10.444, 9.86944, 169.751, 169.638, -10.444, 9.86944, 169.656, -10.444,
 10.425, -9.86944, -169.751, -169.638, 9.88847, 169.656, -10.444, 10.425, -9.86944,
 -169.656, 10.444, -10.425, 10.425, -9.86944, -169.656, 10.444, -9.86944,
 -169.656, 10.444, -10.425, 9.86944, 169.656, -9.88847, -169.751, -169.733, -169.733}
```

```
Tom6 = CDMACoarseSynchroniser[ModOutputUnEncodedFrame3 // Drop[#, 250 4]&,
  Goldseqlist // #[[3]]& , 100, 4]

DeModulator[{{10.3815 - 157.056 I, 157.056 + 10.3815 I, -10.3815 + 157.056 I,
    -157.056 - 10.3815 I, 10.3815 - 157.056 I, 157.056 + 10.3815 I, -10.3815 + 157.056 I,
    -156.862 - 10.62 I, 15.601 + 9.55746 I, -157.176 - 10.2839 I, 15.601 + 9.55746 I,
    -157.176 - 10.2839 I, 15.601 + 9.55746 I, -157.37 - 10.0454 I, 10.3815 - 157.056 I,
    156.862 + 10.62 I, -15.601 - 9.55746 I, 157.37 + 10.0454 I, -10.62 + 156.862 I,
    9.55746 - 15.601 I, -10.2839 + 157.176 I, 9.75774 - 15.3789 I, -15.6783 - 9.3815 I,
    -9.18122 + 15.9004 I, 10.2839 - 157.176 I, -9.75774 + 15.3789 I, 15.9004 + 9.18122 I,
    -157.176 - 10.2839 I, 15.601 + 9.55746 I, -157.37 - 10.0454 I, 10.3815 - 157.056 I,
    156.862 + 10.62 I, -15.3789 - 9.75774 I, -9.3815 + 15.6783 I, 15.6783 + 9.3815 I,
    9.18122 - 15.9004 I, -10.2839 + 157.176 I, 9.75774 - 15.3789 I, -15.6783 - 9.3815 I,
    -9.18122 + 15.9004 I, 10.2839 - 157.176 I, -9.75774 + 15.3789 I, 15.9004 + 9.18122 I,
    -157.176 - 10.2839 I, 15.601 + 9.55746 I, -157.37 - 10.0454 I, 10.3815 - 157.056 I,
    156.862 + 10.62 I, -15.3789 - 9.75774 I, -9.18122 + 15.9004 I, 10.2839 - 157.176 I,
    -9.55746 + 15.601 I, 10.0454 - 157.37 I, 156.862 + 10.62 I, -15.601 - 9.55746 I,
    157.176 + 10.2839 I, -15.601 - 9.55746 I, 157.37 + 10.0454 I, -10.62 + 156.862 I,
    9.55746 - 15.601 I, -10.0454 + 157.37 I, -156.862 - 10.62 I, 15.601 + 9.55746 I,
    -157.176 - 10.2839 I, 15.3789 + 9.75774 I, 9.18122 - 15.9004 I, -10.0454 + 157.37 I,
    -157.056 - 10.3815 I, 10.3815 - 157.056 I, 157.056 + 10.3815 I}, {-7.65342 - 169.733 I,
    169.733 - 7.65342 I, 7.65342 + 169.733 I, -169.733 + 7.65342 I, -7.65342 - 169.733 I,
    169.733 - 7.65342 I, 7.65342 + 169.733 I, -169.638 + 6.41574 I, 17.2252 + 9.88847 I,
    -169.656 + 6.44773 I, 17.2252 + 9.88847 I, -169.656 + 6.44773 I, 17.2252 + 9.88847 I,
    -169.751 + 7.68541 I, -7.65342 - 169.733 I, 169.638 - 6.41574 I, -17.2252 - 9.88847 I,
    169.751 - 7.68541 I, 6.41574 + 169.638 I, 9.88847 - 17.2252 I, 6.44773 + 169.656 I,
    10.444 - 16.382 I, -16.412 - 10.425 I, -9.86944 + 17.2553 I, -6.44773 - 169.656 I,
    -10.444 + 16.382 I, 17.2553 + 9.86944 I, -169.656 + 6.44773 I, 17.2252 + 9.88847 I,
    -169.751 + 7.68541 I, -7.65342 - 169.733 I, 169.638 - 6.41574 I, -16.382 - 10.444 I,
    -10.425 + 16.412 I, 16.412 + 10.425 I, 9.86944 - 17.2553 I, 6.44773 + 169.656 I,
    10.444 - 16.382 I, -16.412 - 10.425 I, -9.86944 + 17.2553 I, -6.44773 - 169.656 I,
    -10.444 + 16.382 I, 17.2553 + 9.86944 I, -169.656 + 6.44773 I, 17.2252 + 9.88847 I,
    -169.751 + 7.68541 I, -7.65342 - 169.733 I, 169.638 - 6.41574 I, -16.382 - 10.444 I,
    -9.86944 + 17.2553 I, -6.44773 - 169.656 I, -9.88847 + 17.2252 I,
    -7.68541 - 169.751 I, 169.638 - 6.41574 I, -17.2252 - 9.88847 I,
    169.656 - 6.44773 I, -17.2252 - 9.88847 I, 169.751 - 7.68541 I, 6.41574 + 169.638 I,
    9.88847 - 17.2252 I, 7.68541 + 169.751 I, -169.638 + 6.41574 I, 17.2252 + 9.88847 I,
    -169.656 + 6.44773 I, 16.382 + 10.444 I, 9.86944 - 17.2553 I, 7.68541 + 169.751 I,
    -169.733 + 7.65342 I, -7.65342 - 169.733 I, 169.733 - 7.65342 I}}]

Tom6[[1, 1]] SeqI // Re

{157.056, 157.056, 157.056, 157.056, 157.056, 157.056, 157.056, 156.862, -9.55746,
 -157.176, 9.55746, 157.176, -9.55746, -157.37, -157.056, -156.862, 9.55746, 157.37,
 156.862, -9.55746, -157.176, 9.75774, -9.3815, 9.18122, 157.176, -9.75774, 9.18122,
 157.176, -9.55746, -157.37, -157.056, -156.862, 9.75774, -9.3815, 9.3815, -9.18122,
 -157.176, 9.75774, -9.3815, 9.18122, 157.176, -9.75774, 9.18122, 157.176, -9.55746,
 -157.37, -157.056, -156.862, 9.75774, -9.18122, -157.176, 9.55746, 157.37,
 156.862, -9.55746, -157.176, 9.55746, 157.37, 156.862, -9.55746, -157.37,
 -156.862, 9.55746, 157.176, -9.75774, 9.18122, 157.37, 157.056, 157.056, 157.056}

Tom6[[1, 1]] SeqI // Im

{-10.0454, -10.3815, -10.3815, -10.3815, -10.3815, -10.3815, -10.3815, -10.62,
 -15.3789, 15.6783, -15.6783, 15.6783, -15.6783, 15.9004, -10.0454, -10.62, -15.3789,
 15.9004, -10.2839, -15.3789, 15.6783, -15.9004, 10.0454, 10.62, 15.3789, -15.9004,
 10.2839, 15.3789, -15.6783, 15.9004, -10.0454, -10.62, -15.601, 10.0454, 10.3815,
 10.62, 15.3789, -15.9004, 10.0454, 10.62, 15.3789, -15.9004, 10.2839,
 15.3789, -15.6783, 15.9004, -10.0454, -10.62, -15.601, 10.2839, 15.3789,
 -15.6783, 15.9004, -10.2839, -15.3789, 15.6783, -15.6783, 15.9004, -10.2839,
 -15.3789, 15.9004, -10.2839, -15.3789, 15.6783, -15.9004, 10.2839, 15.601,
 -10.0454, -10.3815, -10.3815}

Tom6[[1, 2]] SeqI // Re

{-169.751, -169.733, -169.733, -169.733, -169.733, -169.733, -169.733, -169.638,
 10.444, -10.425, 10.425, -10.425, 10.425, -9.86944, -169.751, -169.638, 10.444,
 -9.86944, -169.656, 10.444, -10.425, 9.86944, 169.751, 169.638, -10.444, 9.86944,
 169.656, -10.444, 10.425, -9.86944, -169.751, -169.638, 9.88847, 169.751, 169.733,
 169.638, -10.444, 9.86944, 169.751, 169.638, -10.444, 9.86944, 169.656, -10.444,
 10.425, -9.86944, -169.751, -169.638, 9.88847, 169.656, -10.444, 10.425, -9.86944,
 -169.656, 10.444, -10.425, 10.425, -9.86944, -169.656, 10.444, -9.86944,
 -169.656, 10.444, -10.425, 9.86944, 169.656, -9.88847, -169.751, -169.733, -169.733}
```

```
Table[CDMAPositionFinder[ModOutputUnEncodedFrame3 // Drop[#, 250 4 + i 20]&,
     Goldseqlist // #[[3]]&, 4] // Flatten // Abs // Max,
  {i, 251, 350}]

{22.6392, 22.6392, 22.6392, 17.4452, 17.6558, 17.6558, 17.6558, 17.6558, 17.4676,
 17.4676, 18.088, 18.088, 18.088, 18.088, 19.6342, 19.6342, 19.6342, 19.6342, 19.6342,
 13.457, 16.0791, 16.0791, 16.0791, 16.0791, 19.95, 19.95, 19.95, 19.95, 19.95,
 18.3042, 16.6993, 16.7542, 23.0831, 25.3773, 25.3773, 25.3773, 25.3773, 24.601,
 24.601, 24.601, 18.7221, 20.5255, 26.9209, 26.9209, 26.9209, 26.9209, 18.9705,
 18.9705, 18.9705, 20.3921, 22.6392, 22.6392, 22.6392, 22.6392, 17.4452, 17.6558,
 17.6558, 17.6558, 17.6558, 17.4676, 17.4676, 18.088, 18.088, 18.088, 18.088,
 19.6342, 19.6342, 19.6342, 19.6342, 19.6342, 13.457, 16.0791, 16.0791,
 16.0791, 16.0791, 19.95, 19.95, 19.95, 19.95, 19.95, 18.3042, 16.6993,
 16.7542, 23.0831, 25.3773, 25.3773, 25.3773, 25.3773, 24.601, 24.601, 24.601,
 18.7221, 20.5255, 26.9209, 26.9209, 26.9209, 26.9209, 18.9705, 18.9705, 18.9705}

Max[%]

26.9209

Map[Max, %]

{17.6558, 17.6558, 17.6558, 17.6558, 17.4676, 17.4676, 18.088, 18.088, 18.088, 18.088,
 19.6342, 19.6342, 19.6342, 19.6342, 19.6342, 13.457, 16.0791, 16.0791,
 16.0791, 16.0791, 19.95, 19.95, 19.95, 19.95, 19.95, 18.3042, 16.6993,
 16.7542, 23.0831, 25.3773, 25.3773, 25.3773, 25.3773, 24.601, 24.601, 24.601,
 18.7221, 20.5255, 26.9209, 26.9209, 26.9209, 26.9209, 18.9705, 18.9705,
 18.9705, 20.3921, 22.6392, 22.6392, 22.6392, 22.6392, 17.4452}
```

References

1 "Binary Sequences with Gold-Like Correlation but Larger Linear Span" by Serdar Boztas and P . Vijay Kumar IEEE Trans . on Information Theory Vol 40 No 2, March 1984